Error Handling in Java Servlets

Jeff Offutt
http://www.cs.gmu.edu/~offutt/

SWE 642
Software Engineering for the World Wide Web

sources: Professional Java Server Programming, Patzer, Wrox

Six Major Quality Attributes

1. Reliability : Decreased by unhandled failures; the software crashes more often
2. Usability : Poor error handling decreases usability
3. Security : Failures lead to security holes
4. Availability : Not available when software crashes
5. Scalability : As sites grow, unhandled exceptions are more likely to lead to failures
6. Maintainability : Exception handling increases complexity and is difficult to maintain
Web Site Software Failures

😊 Failures happen:
- Faults in programs
- Invalid input data
- Database problems
- Hardware problems
- Network problems

• Successful web sites must handle errors!
😊 Java provides rich error handling constructs

Error Handling

• Computer Science courses typically ignore error handling
• Advanced software engineering (SWE 619) pays little attention to it
• It’s considered boring!

**Effective** web software engineers pay a lot of attention to error handling
Seven Steps for Error Handling

When errors occur, servlet should:

1. Recognize the problem
2. Diagnose the problem
3. Handle the problem (if possible)
4. Inform the user (if necessary)
5. Inform the administrator (if necessary)
6. Log details of the problem
7. Repeat service (if productively possible)

Error Handling – What & Why

• These are crucial things the engineer should know
• The hard part:

  Anticipating what might go wrong!
Java Exceptions

When a run-time failure is detected:
• An exception is thrown to the program
• The program can handle it
• Programs can also explicitly throw exceptions

Exceptions are objects:

```
java.lang.Exception
    ↓
Exception objects
```

Methods must list the exceptions they can throw (`throws`)

Java Exceptions (2)

• Rule: If a method `p()` calls another method `m()` that throws an exception `e`, `p()` must either:
  – `handle e`, or
  – `throws e` (that is, as part of the method signature)

• Hints (*usability issues* — SWE 632):
  – Try to handle exceptions low, not high
  – Try to handle exceptions without informing users
  – If users enter wrong data, only make them re-enter the data that was wrong
  – Make error messages polite, non-accusing, explicit, and be sure to let users know how to correct the problem

• Never send exceptions back to the user
  – Messages are not intended for users
  – Response object is not sent
Handling Exceptions

Exceptions are thrown and caught

```java
try {
    // statements that can throw exceptions, including method
    // calls, must be in try blocks
} catch (NumberFormatException e) {
    // do stuff ...
}
```

```java
} catch (Exception e) {
    // Generic exception, handles all others
    // from the exception base class.
}
```

Handling Exceptions (2)

Typical exception methods:

- `String toString()` – returns name + error message
  - Of questionable usefulness to user.

- `void printStackTrace()` – sequence of method calls
  - Send to log or sysadm, NEVER to user!

- `String getMessage()` – error message in `toString()`
Exceptions Examples

Look at LoanCalculator example
http://cs.gmu.edu/~offutt/classes/642/examples/servlets/

Logging Error Data

Why send error data to the SysAdmin?

If users make the same mistake repeatedly, the web page and instructions should be re-designed
Logging Error Data (2)

- Use `HttpServletResponse` object
- Two methods can be used:
  1. `sendError(int statusCode)`
  2. `sendError(int statusCode, String message)`
- The `sendError()` methods do two things:
  1. Sets the response status code
  2. “Commits” the response:
     - Puts message in an error log
     - Usually sends an HTML page to the user
     - Message should be user oriented!

Logging Error Data (3)

- HTTP status codes:
  - 400: Syntactically incorrect
  - 401: Requires HTTP authentication
  - 403: The server refused to fulfill request
  - 404: Resource is unavailable
  - 500: An error in the HTTP server
  - 501: Server does not support the functionality needed
  - 503: HTTP server is overloaded
- Using the built-in methods saves trouble and can be more reliable, and saves information into the log
Logging Error Data (4)

You can also put messages into the log without bothering the user:

1. Get a `ServletContext` object from the `ServletConfig` object:
   ```java
   ServletContext sc = getServletContext();
   ```
2. Call the `log()` method:
   ```java
   sc.log("JOServlet: method m1() was executed.");
   ```

This message is sent to a log file on the system: the location of the log file is determined when the web server is installed
- `localhost`: `C:\Tomcat\logs\`
- `apps-swe642`: `/usr/share/tomcat6/logs/`

Servlet Exceptions

- “do-do” methods have “throws ServletException”
- If your servlet throws a `ServletException`, serious things happen:
  - Execution stops
  - Response object is not returned to user
  - Servlet engine takes control and usually:
    - Client gets a cryptic error message (sending the user to another web site!)
    - The error is logged

This is a last resort technique!
Summary
User Responses to Exceptions

The previous discusses how to handle exceptions, but the designer must think carefully about what to do to handle exceptions.

This is properly a subject of SWE 632, User Interface Design …